PATENT
8947-000062/US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Hee-Kwan SON	Conf. No.:	Unknown
Application No.:	10/736,838	Group Art Unit:	Unknown
Filing Date:	December 17, 2003	Examiner:	Unknown

Title: MONTGOMERY MODULAR MULTIPLIER AND METHOD THEREOF

PRIORITY LETTER

February 25, 2004

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sirs:

Pursuant to the provisions of 35 U.S.C. 119, enclosed is/are a certified copy of the following priority document(s).

Application No.	Date Filed	Country
2003-29445	5/9/2003	Korea

In support of Applicant's priority claim, please enter this document into the file.

Respectfully submitted,

HARNESS, DICKEY, & PIERCE, P.L.C.

By _____
John A. Castellano, Reg. No. 35,094

P.O. Box 8910
Reston, Virginia 20195
(703) 668-8000

JAC/cah
Enclosure: 1 certified copy

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2003-0029445
Application Number

출 원 년 월 일 : 2003년 05월 09일
Date of Application MAY 09, 2003

출 원 인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2003 년 12 월 19 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【제출일자】	2003.05.09
【발명의 명칭】	저전력 모듈로 곱셈을 수행하는 연산장치
【발명의 영문명칭】	MODULAR MULTIPLICATION CIRCUIT WITH LOW POWER
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	임창현
【대리인코드】	9-1998-000386-5
【포괄위임등록번호】	1999-007368-2
【대리인】	
【성명】	권혁수
【대리인코드】	9-1999-000370-4
【포괄위임등록번호】	1999-056971-6
【발명자】	
【성명의 국문표기】	손희관
【성명의 영문표기】	SON,HEE-KWAN
【주민등록번호】	690311-1551517
【우편번호】	442-706
【주소】	경기도 수원시 팔달구 망포동 동수원엘지빌리지 106동 1101호
【국적】	KR
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인 임창현 (인) 대리인 권혁수 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	18 면 18,000 원

【우선권주장료】	0 건	0 원
【심사청구료】	13 항	525,000 원
【합계】	572,000 원	
【첨부서류】	1. 요약서·명세서(도면)_1통	

【요약서】

【요약】

본 발명은, 소정주파수의 클럭에 응답하며 제1피연산자와 제2피연산자 및 제3피연산자를 이용하여 소정의 결과값을 누산기를 통하여 구하는 연산장치에 관한 것으로서, 상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하여 상기 누산기로 전송하는 제1경로; 상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하여 상기 누산기로 전송하는 제2경로; 상기 제1경로를 제어하는 제1신호들을 발생하는 제1선택회로; 상기 제2경로를 제어하는 제2신호들을 발생하는 제2선택회로; 그리고 상기 제1신호들이 발생될 때까지 상기 제2신호들을 상기 클럭에 응답하여 소정시간동안 저장하는 수단을 구비한다.

【대표도】

도 1

【색인어】

모듈로 연산, 저전력, 파이프라인

【명세서】

【발명의 명칭】

저전력 모듈로 곱셈을 수행하는 연산장치{MODULAR MULTIPLICATION CIRCUIT WITH LOW POWER}

【도면의 간단한 설명】

도 1은 본 발명에 따른 모듈로 곱셈기의 구성을 보여주는 블럭도이다.

도 2는 도 2에 보인 모듈러스 재부호화기의 내부 구성을 보여주는 회로도이다.

도 3은 도 2에 보인 부스 재부호화기의 내부 구성을 보여주는 회로도이다.

도 4는 도 3 또는 도 4에서 사용된 멀티플렉서의 회로구성의 일례를 보여주는 회로도이다.

도 5는 파이프라인 방식이 적용되지 않은 경우의 처리과정을 보여주는 타이밍도이다.

도 6은 본 발명에 따른 모듈로 곱셈기에서의 파이프라인방식이 적용된 경우의 처리과정을 보여주는 타이밍도이다.

본 발명에 따른 도면들에서 실질적으로 동일한 구성과 기능을 가진 구성요소들에 대하여는 동일한 참조부호를 사용한다.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<8> 본 발명은 데이타 보안을 위한 암호화(cryptography) 시스템에 관한 것으로서, 특히 암호화시스템에서의 모듈로 곱셈을 수행하기 위한 연산장치에 관한 것이다. 본 발명에서는, 연산속도를 증대시키고 전력소모를 줄이는 모듈로 곱셈기를 제공한다.

<9> 컴퓨터 네트워크 또는 유무선통신을 통하여 각종 데이타를 주고받는 정보통신 환경에서는 데이타의 보안성을 유지하기 위한 암호화 시스템의 중요성이 날로 커지고 있다. 특히, 전자결제 또는 인증시스템에서는 부호화(encrytion) 및 복호화(decryption) 기술을 적용하여 보안성을 확보하는 것이 필수적이다. 암호화 기술은 크게 비밀키(대칭키 또는 공통키; secret key, symmetric key, private key, common key) 방식과 공개키(비대칭키; public key, asymmetric key) 방식으로 분류할 수 있다.

<10> 비밀키 방식은 미합중국 상무성의 국립표준국(NBS)에 의한 DES(Data Encryption System) 암호 알고리즘이 대표적인 예이며, 그 외에 구 소련의 GOST(GOvernment STandard), 스위스의 IDEA(International Data Encryption Algorithm) 등이 있다. 비밀키 암호화 방식에서는 정보교환 당사자간에 동일한 비밀키를 공유하여야 하므로 별도의 안전한 채널이 필요하고 특히 여러 사람과의 정보교환시에는 한 사용자가 많은 비밀키 채널을 유지 및 관리하여야 한다는 단점이 있다.

<11> 반면에, 공개키 암호화 방식은 암호화하는 키와 복호화하는 키가 서로 다르기 때문에 둘 중 하나를 알더라도 그 에 대응하는 키를 알기 어렵도록 하는 시스템이다. 공개키 암호화 방식에서는, 하나의 비밀키와 다른 하나의 공개키를 사용하여 정보를 스크램블(scramble)/디스크램블(descramble)한다. 공개키 암호화 방식은, 키를 공유하기 위한 별도의 채널이 필요치 않아 키 관리가 수월하고 비밀키 방식에 비해 보안성은 향상되지만, 두가지의 서로 다른 키를 생성하고 이를 통하여 정보를 복원하기 위한 수학적인 연산과정이 복잡하여 고속처리에 제한이 있는 것으로 알려져 있다.

<12> 공개키 암호화 알고리즘은, 1976년에 Diffie와 Helman에 의해 최초로 제안된 이래, 현재는 RSA(R.L.Rivest, A.Shamir, and L.Adelman; "A method for obtaining digital signatures

and public-key cryptosyatems"; Commun. ACM, Vol.21, pp.120-126, Feb. 1978, 또는 미합중국 특허 4,405,829) 알고리즘이 대표적인 방식으로 알려져 있다. 이 RSA방식은 다른 종류의 공개키 암호화 시스템인 Diffie-Hellman 방식 또는 DSA(Digital Signature Algorithm) 방식과 마찬가지로 모듈로 곱셈(modulo-multiplication)을 기본연산으로 하고 있다.

<13> RSA 알고리즘은 특히 1024비트 이상의 큰 정수(integer)를 기반으로 한 모듈로 연산에 의해 수행되며, 큰 정수 계수(modulus)의 소인수 분해가 매우 어려움에 그 안전성의 근거를 두고 있다.

<14> RSA 암/복호화 알고리즘은 모듈로 멱승 연산(modulo exponentiation operation)을 기본 연산으로 사용하고 있다. 평문(plaintext) P를 암호문(ciphertext) C로 암호화(encryption)하기 위하여 두개의 양의 정수로 구성된 공개키 E(암호화지수 또는 암호화키) 및 M(계수;modulus)을 이용하여 연산식 $C = P^E \bmod M$에 의해 암호문 C를 생성한다. 암호문 C를 평문 P로 복호화(decryption)함에 있어서는 비밀키 D(복호화지수)를 이용하여 연산식 $P = C^D \bmod M$(C를 D번 곱한 값을 M으로 나눈 나머지를 취함)에 의해 평문 P를 구한다. 예들 들면, 앨리스(Alice)라는 사람이 밥(Bob)이란 사람에게 암호화된 메세지를 보내기 위해서는, 모든 사람들에게 공개되어 있는 밥의 공개키 E로써 메세지를 암호화한 뒤 공개채널(unsecure channel)을 통하여 밥에게 메세지(암호문 C)를 보낸다. 그러면, 밥은 자신만이 알고 있는 비밀키 D를 사용해서 암호화된 메세지를 복호화해서 원래의 메세지(평문 P)를 읽게 된다.

<15> 암/복호화를 위한 키를 선택하기 위해서는, 우선 두개의 매우 큰 소수(prime number) p 및 q를 임의로 선택하여 $M = p \times q$를 계산하고, $\gcd[D, (p-1)(q-1)] = 1$을 만족하는 수를 복호화 키 D로 한다. 여기서 gcd는 최대공약수(greatest common divider)이다.암호화 키 E는 $E \times D = 1 \bmod [(p-1)(q-1)]$을 만족하는 암호화키 E를 구한다.

<16> RSA를 위한 모듈로연산(modular multiplication)은 내부에 곱셈과 나눗셈이 복합되어 있어 계산구조가 복잡하고 워드 크기가 통상 1024 비트 이상으로 크기 때문에 고속의 실시간 처리를 위한 하드웨어 모듈의 구현이 어렵다. 즉, 사용하는 키 값이 증가하면 모듈로 멱승을 위한 연산속도가 감소되므로 고속연산을 위해서는 멱승에 필요한 모듈로 곱셈 수를 가능한 최소로 줄여야 한다. RSA 알고리즘의 주된 연산인 모듈로 멱승은 연속된 모듈로 곱셈으로 구성되므로 RSA 암호화 시스템 구현의 관건은 모듈로 곱셈기의 설계에 있다.

<17> RSA 암호화 시스템을 구현하는 방법은 크게 두가지로 나눌 수 있다. 하나는 어레이 곱셈기(array multiplier)를 응용한 것이고, 다른 하나는 1985년에 제안된 몽고메리(Montgomery) 알고리즘(P.L.Montgomery, "Modular Multiplication Without Trial Division", Mathematics of Computation, vol.44, No.170, pp.519-521, 1985)을 이용한 곱셈방법이다.

<18> 몽고메리 알고리즘은 하드웨어 구현이 어려운 임의의 수에 대한 모듈로 연산을 단순히 곱셈 및 덧셈연산과 쉬프트(shift)연산으로 변환하여 해결하기 때문에 디지탈 IC설계를 통한 구현이나 중앙처리장치(CPU) 또는 디지탈신호처리장치(DSP) 등에서의 알고리즘 실현이 용이하도록 하는 장점이 있다. 몽고메리 알고리즘에서는 곱셈연산의 전후에 피연산자 변환(operand transformation) 과정이 필요하다. 따라서, 단일 곱셈연산에서는 다른 일반적인 모듈로곱셈기에 비하여 처리성능이 느릴 수 있지만, RSA 알고리즘과 같이 동일한 모듈러스에 대하여 반복적인 곱셈연산을 행하는 응용영역에서는 매번의 피연산자 변환작업이 필요하지 않기 때문에 다른 모듈로 곱셈기에 비해 고속으로 암호화 작업을 처리할 수 있다.

<19> 통상적인 몽고메리방식의 모듈로 연산 방식에 의하면, 모듈러스 값의 정수배인 모듈러스 곱(multiple of modulus)의 값들과, 피승수(multiplicand)에 승수(multiplier)의 각 디지트

(digit)를 곱한 부분곱(partial product)의 값들을 1의 보수(1's complement number)의 값들로 생성시킨다. 1의 보수화된 모듈러스곱 및 부분곱의 값들은 보상워드(compensation word)와 함께 누산기에 입력되며 보상워드가 반영됨에 의해 모듈러스곱 및 부분곱의 값들로부터 2의 보수(2's complement number)로 표현된 최종 결과값이 구하여 진다.

<20> 이러한 전 과정은 일정한 주파수를 가지는 클럭신호에 동기하여 진행된다. 모듈러스곱과 부분곱을 1의 보수화하여 누산기로 입력하고 최종 결과값을 2의 보수화하는 것은 모듈러스곱 및 부분곱의 정수배에 따른 연산과정의 복잡도를 줄이기 위함이다.

<21> 통상적인 몽고메리 알고리즘의 일종인 디지트-시리얼 인터리브드 몽고메리 곱셈(Digit-Serial Interleaved Montgomery Multiplication; DSIMN) 알고리즘은 다음과 같다.

<22> Stimulus :

<23> Integer modulus M : $M > 2$, and M is odd

<24> Integer M' : $(-M \cdot M') \bmod r = 1$

<25> Integer n : Length of M in bits ($n=kN$)

<26> Integer N : Length of M in digits

<27> Integer r : Radix for the digit system ($r = 2^k$)

<28> Integer k : Length of radix r in bits ($k = \log_2 r$)

<29> Integer R : $R = 2^n = r^N$

<30> Integer R^{-1} : $(R \cdot R^{-1}) \bmod M = 1$

<31> Integer multiplicand A : $0 \leq A < M$

<32> Integer multiplier B : $0 \le B < M$

<33> $$B = \sum_{I=0}^{N-1} b_I \cdot r^I \quad \text{where}$$

<34> Response :

<35> $S^N = A \cdot B \cdot R^{-1} \pmod{M}$ and $0 \le S_N < M$

<36> Method :

<37> S_0 : 0

<38> for I := 0 to (N - 1) do

<39> $q_I := (((S_I + b_I A \bmod r) \cdot M') \bmod r$

<40> $S_I+1 := (S_I + b_I A + q_I M) / r$

<41> endfor

<42> if ($S_N \ge M$) $S_N := S_N - M$

<43> 위 알고리즘에서 몫(quotient) q_I 가 뜻하는 바는 "이전 싸이클까지 누산기(accumulator)에 저장된 값(=S_1)에 현재 싸이클의 부분곱(partial product)의 값(=$b_I A$)을 더해 만든 값(= $S_I + b_I A$)의 하위 k개 bit가 $(0)_2$이 될 수 있도록 더해 줄 모듈러스(modulus) M의 갯수"이다. 잉여수 체계(Residue Number System; RNS)에서 어떤 수에 모듈러스의 정수배를 더한 수는 원래의 수 와 같다. 그러므로 모듈러스 M의 정수배인 $q_I M$을 더한 수는 RNS 에서 원래 수 와 같은 수이

다. 또한 하위 k개 비트를 $(0)_2$ 으로 만든 후 $r(=2^k)$로 나누면(즉, k-bit shift right 하면) 유효자리 내의 숫자는 유지되므로 정보의 유실이 없게 된다.

<44> 위 알고리즘을 하드웨어로 구현하려면 매 싸이클 마다 b_IA (=partial product, PP_I)와 $q_I M$ (multiple of modulus; MM_I)을 구해서 누산기에 인가하여야 한다. 그런데, MM_I의 값 (또는 quotient q_I의 값)을 결정하기 위해서는 먼저 PP_I의 값을 계산한 후 계산된 PP_I의 값과 누산기에 누적되어 있는 S_I의 값을 더한 값을 계산하여야만 한다. 따라서 누산기의 입력값인 PP_I와 MM_I가 동시에 누산기로 인가될 수 없고 PP_I가 먼저 인가된 후 일정 시간이 경과된 후에야 MM_I가 누산기에 인가 된다. 이런 특성 때문에 누산기가 출력값의 갱신을 위한 논리계산(logic calculation)을 두번 (PP_I가 인가될 때 한번, MM_I가 인가될 때 또 한번) 수행하게 되고 이는 결국 불필요하게 동적 전력소모(dynamic power consumption)를 높이게 된다.

<45> 또한 매 싸이클 마다 PP_I와 MM_I의 값이 다단계 논리회로(multi-level logic circuit)를 거쳐서 만들어지므로 해당 싸이클에서 PP_I와 MM_I의 값이 최종값까지 도달해가는 과정에서 많은 수의 스퓨러스 트랜지션(spurious transition 또는 glitch)이 발생되고 결국 이런 스퓨러스 트랜지션에 의해 불필요한 동적 전력소모가 유발된다.

<46> 그리고 위 알고리즘은 한 싸이클 내에서 PP_I와 MM_I의 값을 계산하는 과정과 그 값들을 누산기를 통해 누적하는 과정이 모두 이루어지고 있으므로 신호경로(critical path)가 길게 형성되고 따라서 처리성능(processing performance)을 높이기 위해 개선이 필요하다.

<47> 본 발명은 모듈로 곱셈장치를 위에서 열거한 바와 같은 소모전력과 처리성능 측면에서 개선하는 방안에 대한 것이다.

【발명이 이루고자 하는 기술적 과제】

<48> 따라서, 본 발명의 목적은 저전력소모에 적합한 모듈로 곱셈 연산장치를 제공함에 있다.

<49> 본 발명의 다른 목적은 낮은 전력소모로써 동작속도를 향상시킬 수 있는 모듈로 곱셈 연산장치를 제공함에 있다.

<50> 본 발명은, 소정주파수의 클럭에 응답하며 제1피연산자와 제2피연산자 및 제3피연산자를 이용하여 소정의 결과값을 누산기를 통하여 구하는 연산장치에 있어서: 상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하여 상기 누산기로 전송하는 제1경로; 상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하여 상기 누산기로 전송하는 제2경로; 상기 제1경로를 제어하는 제1신호들을 발생하는 제1선택회로; 상기 제2경로를 제어하는 제2신호들을 발생하는 제2선택회로; 그리고 상기 제1신호들이 발생될 때까지 상기 제2신호들을 상기 클럭에 응답하여 소정시간동안 저장하는 수단을 구비한다.

<51> 상기 제1신호들과 상기 제2신호들은 상기 클럭에 응답하여 상기 제1경로와 상기 제2경로에 동시에 각각 인가된다.

<52> 상기 제1경로는, 상기 제1신호들 중 하나에 응답하여 상기 제1연산값들 중 하나를 선택하는 제1멀티플렉서와, 상기 제1신호들 중 다른 하나에 응답하여 상기 제1멀티플렉서의 출력을 상기 누산기로 전송하는 제1게이트회로를 포함한다. 상기 제2경로는, 상기 제2신호들 중 하나에 응답하여 상기 제2연산값들 중 하나를 선택하는 제2멀티플렉서와, 상기 제2신호들 중 다른 하나에 응답하여 상기 제2멀티플렉서의 출력을 상기 누산기로 전송하는 제2게이트회로를 포함한다.

<53> 상기 제2신호들 중 상기 하나의 신호는 상기 클럭의 싸이클마다 평균 해밍 디스턴스가 최소인 복수의 비트들로 구성된다. 상기 제1신호들 및 제2신호들은 상기 클럭에 응답하여 파이프라인 방식으로 구동되며, 상기 제1 및 제2연산값들이 상기 제1 및 제2신호들에 각각 응답하여 파이프라인 방식으로 구동된다.

【발명의 구성 및 작용】

<54> 본 발명의 실시예에서 보이는 곱셈장치의 논리연산체계는 공개키 방식의 암호화알고리즘을 적용하는 컴퓨터 시스템 또는 통신망에 적용될 수 있으며, 또한 휴대가능한 집적회로 카드(또는 스마트카드)에 내장되어 운용될 수 있다.

<55> 본 발명에 따른 실시예는 적어도 1024비트 이상의 큰 정수를 기반으로 하는 모듈로 곱셈 연산에 적용될 수 있다.

<56> 본 발명은 모든 종류의 모듈로 곱셈 연산장치에 사용될 수 있지만 편의상 radix-4 구조이고 몽고메리 알고리즘을 사용하는 모듈로 곱셈 연산장치를 기준으로 설명하기로 한다.

<57> 즉, 본 발명의 실시예에서는, 모듈러스의 길이가 n이고 곱셈연산을 위한 누산은 클럭신호의 싸이클에 따라 디지트(N) 단위로 수행된다. 여기서, n = 2N이다. 예컨대 n이 1024라면 N은 512가 된다. 본 발명의 실시예에 관한 설명 또는 도면들에서 문자부호 "I"는 0 ~ N-1 중 하나의 디지트 순서값을 나타낸다.

<58> 도 1은 본 발명에 따른 모듈로곱셈 연산장치의 구성을 보여 준다.

<59> 도 1의 모듈로곱셈 연산장치는 모듈러스 레지스터 M_REG, 피승수 레지스터 A_REG, 승수 레지스터 B_REG, 4:1 멀티플렉서들 13 및 23, 모듈러스 재부호화기 70, 부스 재부호화기 80,

앤드게이트회로들 43 및 45, 반가산기 47, 컴프레서 네트워크 50, 누산기 100, 덧셈기들 53 및 57, 플립플럽들 61~65, 그리고 래치들 66~69를 포함한다.

<60> 누산기 100은 컴프레서 네트워크 50과 캐리 레지스터 C_REG 및 섬 레지스터 S_REG으로 구성된다. 누산기 100으로부터 최종결과값 $S_N[n:0]$이 산출된다.

<61> 여기서, 레지스터들 A_REG, B_REG, M_REG, C_REG 및 S_REG와 플립플럽들 61~65는 클럭 CK의 상승 에지(rising edge)에 응답하여 동작하고, 래치들 66~69는 클럭 CK가 로우레벨인 동안에 입력을 통과시킨다.

<62> 레지스터들 M_REG, A_REG, C_REG 및 S_REG는 병렬 입출력형(PIPO; Parallel-In parallel-Out) 레지스터들이다. 레지스터들 M_REG, A_REG, C_REG 및 S_REG는 N+1(= (n/2)+1) 회의 반복연산이 진행되어 최종 결과값 $S_N[n:0]$이 산출될 때까지 클럭 CK의 매 싸이클에 응답하여 동작한다.

<63> 모듈러스 레지스터 M_REG는 n비트로 된 모듈러스 M[n-1:0]을 저장한다. 몽고메리 알고리즘에 따르면 모듈러스 M은 항상 홀수이므로 모듈러스 레지스터 M_REG의 마지막 비트위치(LSB)는 항상 "1"로 설정되어 있다. 또한, 모듈러스 M은 항상 양수이므로 부호비트를 포함하지 않는다. 피승수 레지스터 A_REG는 n+1비트의 길이로 되어 있으며 피승수 A[n:0]를 저장한다. 피승수 레지스터 A_REG의 길이가 n+1인 것은, 피승수 A가 양수와 음수 모두 가능하므로 부호비트(n번째, 즉 최상위 비트 S_A)를 포함시켜야 하기 때문이다. 승수 레지스터 B_REG는 병렬-입력/직렬-출력(PISO; Parallel-In Serial-Out) 레지스터이며 승수 B를 저장한다. 승수 B에는 부호비트와 짝수의 비트길이를 맞추기 위한 비트가 포함되어야 하므로, 승수레지스터 B_REG는 n+2의 비트길이로 되어 있으며 최상위 비트인 n번째 비트와 n+1번째 비트가 부호(S_B)

를 나타낸다. 승수 레지스터 B_REG는 매 클럭싸이클마다 2비트씩 우측 쉬프트동작을 수행한다. 승수 레지스터 B_REG는 매 클럭싸이클마다 우측 쉬프트후에 하위 2비트 b_1 및 b_0와 이전 싸이클의 비트 b_R을 부스 재부호화기 80으로 제공한다.

<64> 모듈러스 레지스터 M_REG로부터 제공되는 모듈러스 M과 모듈러스 M의 1의 보수값 $\underline{M}$을 포함하여 모듈러스곱 MM_I의 값들($\underline{M}$, 0, M, 2M)이 4:1 멀티플렉서 13에 인가된다. 한편, 피승수 레지스터 A_REG로부터 제공되는 피승수 A과 피승수 A의 1의 보수값 $\underline{A}$를 포함하여 부분곱 PP_I의 값들($\underline{2A}$, $\underline{A}$, 0, A, 2A)이 4:1 멀티플렉서 23에 인가된다. 멀티플렉서들 13 및 23에 전달된 모듈러스곱 및 부분곱의 값들은 1의 보수값들($\underline{M}$, $\underline{A}$, $\underline{2A}$)을 포함하고 있으며, 멀티플렉서들 13 및 23에 의해 선택된 후에 누산기 50에서 보상워드 $CW_I[1:0]$이 반영된 연산을 통하여 2의 보수화될 것이다.

<65> 모듈러스 재부호화기 70은, 모듈러스곱 MM_I의 값을 선택하기 위한 조합논리회로이다. 모듈러스 재부호화기 70의 입력은, 멀티플렉서 13을 제어하기 위한 신호 SEL_MM[1:0]로부터 플립플럽 61을 통하여 한 클럭싸이클만큼 지연되어 궤환되는 신호 SEL_MM_D[1:0], 모듈러스 레지스터 M_REG로부터 제공되는 모듈러스 M의 하위 2번째 비트 M[1], 부분곱의 하위 2비트 $SPP_I[1:0]$와 섬워드의 하위 2비트 $S_I[1:0]$의 합이 되는 부분누산합의 하위 2비트 $SPP_I[1:0]$이다.

<66> 모듈러스 재부호화기 70의 출력은, 멀티플렉서 13을 제어하여 모듈러스곱 MM_I의 값들 중 하나를 선택하기 위한 신호 SEL_MM[1:0], 멀티플렉서 13의 출력이 누산기 100의 5-2 컴프레서 네트워크 50으로 전송되는 것을 통제하는 앤드게이트회로 43을 제어하기 위한 신호 EN_MM, 그리고 반가산기 47로 인가되어 선택된 모듈러스곱 MM_I의 비트반전 여부를 알려주기 위한 제어신호 NEG_MM이다.

<67> 여기서, 모듈러스 재부호화기 70의 출력들 중에서, 제어신호 SEL_MM[1:0]은 플립플럽 61을 통하여 한 클럭싸이클만큼 지연되어 궤환입력되는 한편 클럭 CK가 로우레벨인 때에 래치 66을 통하여 멀티플렉서 13으로 인가된다. 또한, 제어신호 EN_MM은 클럭 CK가 로우레벨인 때에 래치 67을 통하여 앤드게이트회로 43으로 인가된다. 모듈러스 재부호화기 70에 관련된 상기 플립플럽 61과 래치들 66 및 67은, 모듈러스곱의 값과 부분곱의 값의 발생시점이 서로 다름에 따른 전력소모의 문제 또는 동작속도상의 한계를 극복하는데 기여한다.

<68> 아래의 표 1은 모듈러스 재부호화기 70의 진리표(truth table)로서, 입출력 값에 따라 멀티플렉서 13에 입력된 모듈러스곱 MM_I의 값들(M, 0, M, 2M; 또는 -1M, 0, +1M, +2M) 중에서 선택되는 모듈러스곱 MM_I의 값을 나타낸다.

<69> 【표 1】

입 력			출 력			선택된 MM_I [n+1:0]
M1	SPP_I [1]	SPP_I [0]	SEL_MM[1:0]	EN_MM	NEG_MM	
0	0	0	SEL_MM_D[1:0]	0	0	0
0	0	1	10	1	1	-1M
0	1	0	11	1	0	+2M
0	1	1	00	1	0	+1M
1	0	0	SEL_MM_D[1:0]	0	0	0
1	0	1	00	1	0	+1M
1	1	0	11	1	0	+2M
1	1	1	10	1	1	-1M

<70> 모듈러스 재부호화기 70에 의해 선택된 모듈러스곱 MM_I의 값은 모듈러스 M의 2배수를 위한 비트와 부호비트를 포함하는 n+2 비트의 길이를 가지며 5-2 컴프레서네트워크 50으로 입력

된다. 모듈러스 재부호화기 70에 의해 모듈러스곱의 값이 선택되고 또한 누산기 100의 컴프레서 네트워크 50으로 전달되는 과정에 관한 보다 구체적인 설명은 후술될 것이다.

<71> 도 2는 표 1에 따라 설계된 모듈러스 재부호화기 70의 상세회로의 일례를 보여 준다. 도 2를 참조하면, 모듈러스 재부호화기 70은, 8개의 입력터미널 E_0~E_7과 3개의 제어터미널 G_0~G_2 및 1개의 출력터미널 Z로 이루어진 8:1 멀티플렉서들 71 및 72, 제어신호 NEG_MM을 출력하는 논리회로 73, 그리고 제어신호 EN_MM을 출력하는 오아게이트 OR1로 구성된다. 논리회로 73은, M1 및 $SPP_I[1]$를 입력하는 익스클루시브 노아게이트 XN1, 그리고 익스클루시브 노아게이트 XN1의 출력과 $SPP_I[0]$를 입력하여 제어신호 NEG_MM을 출력하는 앤드게이트 AN1로 구성된다. 오아게이트 OR1은 $SPP_I[1]$ 및 $SPP_I[0]$을 입력하여 제어신호 EN_MM을 출력한다.

<72> 8:1 멀티플렉서들 71 및 72로부터는 각각 제어신호 SEL_MM[1:0]의 상위비트 SEL_MM[1] 및 하위비트 SEL_MM[0]가 각각 출력된다. 8:1 멀티플렉서들 71 및 72의 제어터미널 G_2, G_1 및 G_0에는 각각 M1, $SPP_I[1]$ 및 $SPP_I[0]$가 공통으로 인가된다. 8:1 멀티플렉서 71의 입력터미널 E_0, E_1, E_2, E_3, E_4, E_5, E_6 및 E_7에는 각각 SEL_MM_D[1], 0, 1, 0, SEL_MM_D[1], 0, 1 및 0이 입력된다. 8:1 멀티플렉서 72의 입력터미널 E_0, E_1, E_2, E_3, E_4, E_5, E_6 및 E_7에는 각각 SEL_MM_D[0], 0, 1, 0, SEL_MM_D[0], 0, 1 및 0이 입력된다.

<73> 도 2에 보인 8:1 멀티플렉서(71 또는 72)는 8개의 입력터미널들 및 3개의 제어터미널에 인가되는 신호들의 값에 따라 그 출력값이 달라질 수 있으나, 도 4에 그 일례를 보인 바와 같이, 앤드게이트들 AN11~AN18과 오아게이트 OR11로써 간단히 구성할 수 있다. 아래의 식 1은 8:1 멀티플렉서의 불리언(Boolean) 방정식을 나타낸다.

<74> $Z = G_2 \cdot G_1 \cdot G_0 \cdot E_7 + G_2 \cdot G_1 \cdot G_0B \cdot E_6$

<75> $+ G_2 \cdot G_1B \cdot G_0 \cdot E_5 + G_2 \cdot G_1B \cdot G_0B \cdot E_4$

<76> $+ G_2B \cdot G_1 \cdot G_0 \cdot E_3 + G_2B \cdot G_1 \cdot G_0B \cdot E_2$

<77> $+ G_2B \cdot G_1B \cdot G_0 \cdot E_1 + G_2B \cdot G_1B \cdot G_0B \cdot E_0$ 식 1

<78> 식 1에서 G_0B~G_2B는 G_0~G_2의 반전비트를 나타낸다.

<79> 도 2의 논리회로 73은, M1과 $SPP_I[1]$을 입력하는 익스클루시브 오아게이트 XR1과, 익스클루시브 노아게이트 XN1의 출력과 $SPP_I[0]$을 입력하여 제어신호 NEG_MM을 발생하는 앤드게이트 AN1으로 구성된다.

<80> 한편, 다시 도 1을 참조하면, 부스 재부호화기 80은, 부분곱 PP_I의 값을 선택하기 위한 조합논리회로이다. 부스 재부호화기 80의 입력은, 멀티플렉서 23에 전송된 부분곱 PP_I의 값들 중 하나를 선택하기 위한 신호 SEL_PP[1:0]가 플립플럽 62를 통하여 한 클럭싸이클만큼 지연되어 궤환된 신호 SEL_PP_D[1:0], 피승수 A의 하위 2비트 A[1:0], 그리고 승수 B의 하위 2비트 b_1 및 b_0와 2비트만큼 우측 쉬프트된 비트 b_R이다. 우측 쉬프트된 비트 b_R은 플립플럽 FF를 통하여 부스 재부호화기 80으로 공급된다. 부스 재부호화기 80으로 피승수 A의 하위 2비트 A[1:0]이 입력되는 것은 부스 재부호화기 80으로부터 부분곱의 하위 2비트 $PP_I[1:0]$를 생성하기 위함이다.

<81> 부스 재부호화기 80의 출력은, 멀티플렉서 23에 전송된 부분곱 PP_I의 값들 중 하나를 선택하기 위하여 멀티플렉서 23으로 인가될 제어신호 SEL_PP[1:0], 멀티플렉서 23의 출력이 누산기 100의 5-2 컴프레서 네트워트 50으로 전송되는 것을 통제하는 앤드게이트회로 45를 제어하

기 위한 신호 EN_PP, 선택된 부분곱 PP_I의 비트반전 여부를 결정하기 위하여 반가산기 47로 인가될 제어신호 NEG_PP, 그리고 덧셈기 57에서 섬워드의 하위 2비트 $S_I[1:0]$과 더해질 부분곱의 하위 2비트 $PP_I[1:0]$이다.

<82> 여기서, 제어신호 SEL_PP[1:0]는 플립플럽 62와 래치 68을 통하여 4:1 멀티플렉서 23에 인가된다. 제어신호 EN_PP는 플립플럽 63과 래치 69를 통하여 앤드게이트회로 45에 인가된다. 제어신호 NEG_PP는 플립플럽 64를 통하여 반가산기 47에 인가된다. 또한, 부분곱의 하위 2비트 $PP_I[1:0]$은 플립플럽 65를 통하여 덧셈기 57에 인가된다. 부스 재부호화기 80에 관련된 상기 플립플럽들 62~64와 래치들 68 및 69는, 모듈러스곱의 값과 부분곱의 값의 발생시점이 서로 다름에 따른 전력소모의 문제 또는 동작속도상의 한계를 극복하는데 기여한다.

<83> 아래의 표 2는 부스 재부호화기 80의 입출력 관계를 보여주는 진리표로서, 선택되는 부분곱 PP_I의 값들(2A, A, 0, A, 2A; 또는 -2A, -1A, 0, +A, +2A)과 부분곱의 하위 2비트 $PP_I[1:0]$의 값들을 나타낸다.

<84> 【표 2】

입 력			출 력				선택된 PP_I [n+1:0]
b_1	b_0	b_R	SEL_PP[1:0]	EN_PP	NEG_PP	PP_I [1:0]	
0	0	0	SEL_PP_D[1:0]	0	0	00	0
0	0	1	00	1	0	{A[1],A[0]}	+1A
0	1	0	00	1	0	{A[1],A[0]}	+1A
0	1	1	11	1	0	{A[0],0}	+2A
1	0	0	01	1	1	{A[0],0}	-2A
1	0	1	10	1	1	{A[1]^A[0],A[0]}	-1A
1	1	0	10	1	1	{A[1]^A[0],A[0]}	-1A
1	1	1	SEL_PP_D[1:0]	0	0	00	0

<85> 제어신호 NEG_PP는 모듈러스 재부호화기 70로부터 제공되는 제어신호 NEG_MM과 함께 반가산기 47로 인가되어 2비트의 보상워드 $CW_I[1:0]$가 누산기 100의 컴프레서 네트워크 50으로 입력되도록 한다. 부스 재부호화기 80에 의해 부분곱의 값이 선택되고 또한 누산기 100의 컴프레서 네트워크 50으로 전달되는 과정에 관한 보다 구체적인 설명은 후술될 것이다.

<86> 도 3은 표 2에 따라 설계된 부스 재부호화기 80의 상세회로의 일례를 보여 준다. 도 3을 참조하면, 부스 재부호화기 80은, 8개의 입력터미널 E_0~E_7과 3개의 제어터미널 G_0~G_2 및 1개의 출력터미널 Z로 이루어진 4개의 8:1 멀티플렉서들 81~84, 제어신호 EN_PP를 출력하는 논리회로 85, 그리고 제어신호 NEG_PP를 출력하는 논리회로 86으로 구성된다. 8:1 멀티플렉서들 81~84의 기본적인 회로구성은 도 4에 보인 것과 동일하다.

<87> 4개의 8:1 멀티플렉서들 81~84로부터 각각 멀티플렉서 23에 인가되는 제어신호 SEL_PP[1:0]의 상위비트 SEL_PP[1], 제어신호 SEL_PP[1:0]의 하위비트 SEL_PP[0], 부분곱의 하위 두번째 비트 $PP_I[1]$, 부분곱의 하위 첫번째 비트 $PP_I[0]$이 각각 출력된다. 8:1 멀티플렉서들 81~84의 제어터미널 G_2, G_1 및 G_0에는 각각 b_1, b_0 및 b_R이 공통으로 인가된다. 8:1 멀티플렉서 81의 입력터미널 E_0, E_1, E_2, E_3, E_4, E_5, E_6 및 E_7에는 각각 SEL_PP_D[1], 0, 0, 1, 0, 1, 1 및 SEL_PP_D[1]이 배정된다. 8:1 멀티플렉서 82의 입력터미널 E_0, E_1, E_2, E_3, E_4, E_5, E_6 및 E_7에는 각각 SEL_PP_D[0], 0, 0, 1, 1, 0, 0 및 SEL_PP_D[0]이 배정된다. 8:1 멀티플렉서 83의 입력터미널 E_0, E_1, E_2, E_3, E_4, E_5, E_6 및 E_7에는 각각 0, A[1], A[1], A[0], A[0]. A[1]^A[0], A[1]^A[0] 및 0이 배정된다. 입력값 A[1]^A[0]는 A[1] 및 A[0]이 익스클루시브 오아게이트 XR1을 통하여 만들어진 값이다. 8:1 멀티플렉서 84의 입력터미널 E_0, E_1, E_2, E_3, E_4, E_5, E_6 및 E_7에는 각각 0, A[0], A[0], 0, 0, A[0], A[0] 및 0이 배정된다.

<88> 제어신호 EN_PP를 출력하는 논리회로 85는, b_1, b_0 및 b_R을 공통으로 입력하는 낸드게이트들 ND3 및 ND4와, 낸드게이트들 ND3 및 ND4의 출력을 입력하여 제어신호 EN_PP를 발생하는 앤드게이트 AN2로 구성된다. 제어신호 NEG_PP를 발생하는 논리회로 86은, b_0 및 b_R을 입력하는 낸드게이트 ND5와, 낸드게이트 ND5의 출력과 b_1을 입력하여 제어신호 NEG_PP를 발생하는 앤드게이트 AN3으로 구성된다.

<89> 기본적으로, 도 1에서, 누산기 100에 입력된 모듈러스곱 $MM_I[n+1:0]$과 부분곱 $PP_I[n+1:0]$은 컴프레서 네트워크 50에서 캐리-세이브 모드로 동작하는 주연산과정과, 캐리레지스터 C_REG 및 섬레지스터 S_REG에 각각 저장된 캐리워드와 섬워드를 합산하여 그 결과를 다시 섬레지스터 S_REG에 저장하는 후변환과정(캐리-전파 모드)을 통하여 최종값 $S_N[n:0]$이 생성된다. 캐리-세이브 동작모드의 주연산과정에서 캐리전파모드의 후변환과정으로의 전환은, 컴프레서 네트워크 50을 제어하는 신호 SW가 "0"에서 "1"로 천이됨에 따른다. 또한, 모듈러스곱 MM_I의 선택에 참여하는 부분누산합의 하위 2비트 $SPP_I[1:0]$를 만들기 위하여, 캐리레지스터 C_REG와 섬레지스터 S_REG로부터 추출된 하위 2비트의 캐리워드 $C_I[1:0]$ 및 섬워드 $S_I[1:0]$는 덧셈기 53을 통하여 합산된 다음($S_I[1:0]$), 그 합산된 결과인 2비트의 섬워드 $S_I[1:0]$는 다시 덧셈기 57에서 부분곱의 하위 2비트 $PP_I[1:0]$과 합산된다. 누산기 100에서, S_CUR[n:0] 및 C_CUR[n+1:0]는 현재 처리되는 섬워드와 캐리워드를 각각 가리키며, S_NXT[n:0] 및 C_NXT[n+1:0]는 n+1회의 반복연산과정에서 다음에 처리될 섬워드와 캐리워드를 각각 가리킨다. 덧셈기 57로부터 생성된 2비트의 부분누산합 $SPP_I[1:0]$은 모듈러스 재부호화기 70으로 인가된다.

<90> 이상의 구성과 기본적인 동작상태에 관한 설명을 참조하여, 본 발명에 따라 모듈러스곱과 부분곱의 값들이 누산기 100에 도달되는 시점을 일치시킴으로써 불피요한 신호천이 또는 글리치현상으로 인한 전력소모를 방지하는 과정에 관하여 설명한다.

<91> 도 1에 보인 본 발명의 모듈로곱셈 연산장치는, 클럭 CK의 상승에지에 응답하는 플립플럽들 61~65의 동작시점을 기준으로 전과정("부스 재부호화기 루우프")과 후과정("모듈러스 재부호화기 루우프")이 시간적으로 동시에 수행되는 2-스테이지 파이프라인(2-stage pipleline) 동작을 수행한다. 2-스테이지 파이프라인 동작에 관하여는 도 6에 도시되어 있으며, 이하 참조된다.

<92> 여기서, "부스 재부호화기 루우프"는 부분곱 PP_I의 값이 선택되어 누산기 100에 입력되는 경로에 해당한다. "모듈러스 재부호화기 루우프"는 누산기 100을 통하여 생성된 2비트의 섬워드 $S_I[1:0]$와 부분곱의 하위 2비트 $PP_I[1:0]$이 합산되어 생성된 부분누산합 $SPP_I[1:0]$이 모듈러스 재부호화기 70으로 입력된 후 모듈러스곱의 값이 선택되어 누산기 100에 입력되는 경로를 포함한 궤환경로이다.

<93> 그러므로, 전과정에 해당하는 "부스 재부호화기 루우프"는 궤환경로에서 제외되어 후과정보다 짧은 시간에 수행됨에 따라 클럭의 동작주파수를 높일 수 있다. 이는 도 1에 보인 플립플럽들 61~65이 이전 싸이클의 해당하는 입력값들을 저장하고 있기 때문에 가능하다. 즉, 플립플럽들 61~65가 이전 사이클의 해당하는 입력값들을 저장하고 있는 동안, 전과정에 속하는 연산과정이 동시에 진행된다.

<94> 예를 들면, 부스 재부호화기 80으로부터 제공되는 제어신호 SEL_PP[1:0]는 임의의 클럭싸이클에서 클럭 CK의 상승에지에 응답하여 플립플럽 62에 저장된다. 일단의 저장된 SEL_PP[1:0]는 새로운 클럭싸이클에서 새로운 값이 제공되지 않는 한 이전 상태를 그대로 유지한다. 마찬가지로, 플립플럽 63이 이전상태의 EN_PP의 값을 저장하고 있는 클럭싸이클 동안, 전술한 전과정을 통하여 부분누산합의 하위 2비트 $SPP_I[1:0]$이 생성되어 EN_MM이 발생된다.

또한, 플립플럽 64가 이전상태의 NEG_PP의 값을 저장하고 있는 클럭싸이클 동안, 전술한 전과정을 통하여 부분누산합의 하위 2비트 $SPP_I[1:0]$이 생성되어 NEG_MM이 발생된다.

<95> 본 발명에 따른 2-스테이지 파이프라인 동작 타이밍을 보여주는 도 6을 참조하면, 매 클럭마다 부스 재부호화기 80으로 인가되는 디지트 단위의 승수값들 B_LSD0~B_LSDN에 응답하여 두번째의 클럭 1로부터 클럭 N+1에 이르기까지 부분곱의 값들 PP0~PPN과 모듈러스곱의 값들 MM0~MMN이 생성됨을 볼 수 있다.

<96> 이를 파이프라인 방식이 적용되지 않은 도 5와 비교하면, 도 6에서 부분곱 및 모듈러스곱의 값들이 생성되는 시점이 도 5의 경우에 비하여 약 한 클럭싸이클만큼 늦고 부분곱 및 모듈러스곱의 값들의 생성이 완료되는 클럭싸이클 수에 있어서 도 6이 N+3인 반면 도 5는 N+2이지만, 도 6에 보인 최소 클럭싸이클 주기 T2는 일반적인 반도체 공정에서 도 5의 최소 클럭싸이클 주기 T1의 약 0.7배로서 T1보다 짧다. 따라서, 그에 따른 동작상의 성능이득은 약 1.4배(≒(10/7)*((N+2)/(N+3)))이다. 또한, N = 512(모듈러스 M의 비트길이가 1024인 경우)이고 T1이 10ns라고 가정하면, 총 싸이클타임에 있어서도 도 6의 본 발명의 경우가 514싸이클 * 7ns로서 이는 도 5의 경우인 513싸이클 * 10ns보다 짧음을 알 수 있다.

<97> 한편, 4:1 멀티플렉서 13에서의 모듈러스곱의 값을 선택하기 위한 제어신호 SEL_MM[1:0]과 4:1 멀티플렉서 23에서의 부분곱의 값을 선택하기 위한 제어신호 SEL_PP[1:0]이 각각 래치 66 및 플립플롭 62에서 클럭이 로우상태인 동안 래치되어 있기 때문에, 멀티플렉서들 13 및 23으로부터의 출력이 새로운 값으로 갱신되는 시점이 일치된다.

<98> 즉, 부스 재부호화기 80으로부터 제공되는 제어신호 SEL_PP[1:0]가 플립플럽 62을 통하여 래치 68에 저장되어 있는 동안, 부분누산합의 하위 2비트 $SPP_I[1:0]$이 모듈러스 재부호화기

70의 8:1 멀티플렉서들 71 및 72에 전송된 다음 제어신호 SEL_MM[1:0]이 발생되어 래치 66에 저장된다. 그리고, 클럭 CK가 로우레벨인 때에 SEL_MM[1:0]과 SEL_PP[1:0]가 동시에 멀티플렉서 13 및 23에 인가됨으로써, 멀티플렉서들 13 및 23의 출력 갱신시점이 동기화되는 것이다.

<99> 이와 아울러, 멀티플렉서 13의 출력의 누산기 100으로의 전송을 통제하는 앤드게이트회로 43을 제어하기 위한 신호 EN_MM과, 멀티플렉서 23의 출력의 누산기 100으로의 전송을 통제하는 앤드게이트회로 45를 제어하기 위한 신호 EN_PP에 대하여도, 전술한 SEL_MM[1:0] 및 SEL_PP[1:0]의 경우와 마찬가지로, 래치들 67 및 69를 통하여 앤드게이트회로들 43 및 45로의 인가시점을 동기화시킨다. 즉, 모듈러스 재부호화기로부터 발생되는 제어신호 EN_MM도 후과정에 속하는 부분누산합의 하위 2비트 $SPP_I[1:0]$에 응답하여 출력되기 때문에, 부스 재부호화기 80으로부터 출력되는 전과정에 속하는 제어신호 EN_PP는 EN_MM이 발생되기까지 래치 69에서 대기하다가 앤드게이트회로들 43 및 45로 동시에 인가된다.

<100> 결과적으로, 멀티플렉서들 13 및 23로부터 누산기 100까지의 경로에 걸쳐서, 모듈러스곱 $MM_I[n+1:0]$과 부분곱 $PP_I[n+1:0]$의 값들이 매 경로 단계마다 갱신되는 시점을 일치시킴에 따라, 인가 시점의 불일치로 인하여 발생되는 불필요한 신호천이 또는 글리치에 의해 소모되는 동작상의 전력을 줄일 수 있다.

<101> 한편, 본 발명에서는, 매 클럭마다 사용하였던 SEL_MM[1:0]과 SEL_PP[1:0]을 플립플럽 61 및 62에 각각 저장해 두고, 현재 클럭싸이클의 모듈러스곱의 값 또는 부분곱의 값이 "0"으로 선택되어야 하는 경우(표 1 또는 표 2 참조)에, 이전 싸이클의 SEL_MM[1:0]의 값 SEL_MM_D[1:0] 또는 이전 싸이클의 SEL_PP[1:0]의 값 SEL_PP_D[1:0]을 이용한다. 따라서, 현재 클럭싸이클의 모듈러스곱의 값 또는 부

분곱의 값이 "0"으로 선택되어야 하는 경우에, SEL_MM[1:0] 또는 SEL_PP[1:0]의 값이 불필요하게 변하는 것을 방지하여 동작상의 전력소모를 줄인다. 즉, 이 경우에는, 전술한 표 1 또는 표 2에 보인 바와 같이, 앤드게이트회로 43 및 45를 제어하는 신호 EN_MM 또는 EN_PP가 "0"이기 때문에, SEL_MM[1:0] 또는 SEL_PP[1:0]의 값에 상관없이 누산기 100으로 입력되는 모듈러스 곱 또는 부분곱의 값이 "0"으로 된다.

<102> 다른 한편으로, 본 발명에서는, 인접한 싸이클에서 SEL_PP[1:0]이 갖는 평균 해밍 디스턴스(hamming distance; 같은 비트 수를 갖는 2진 부호 사이에 대응되는 비트값이 일치하지 않는 것의 개수)가 최소값을 갖도록 부호화함으로써 동작상의 전력소모를 줄일 수 있다. 이를 설명하면 다음과 같다.

<103> 부스 재부호화기 80에 의해 I번째 싸이클에서 선택된 부분곱 PP_I와 I+1번째 싸이클에서 선택된 부분곱 PP_{I+1}사이에는 다음의 식 2와 같은 관계가 존재한다.

<104> $PP_I = +A$ 또는 $+2A \rightarrow PP_{I+1} \neq +2A$ (즉, $PP_{I+1} \in \{-2A, -A, 0, +A\}$)

<105> $PP_I = -A$ 또는 $-2A \rightarrow PP_{I+1} \neq -2A$ (즉, $PP_{I+1} \in \{-A, +A, 0, +2A\}$) 식 2

<106> 즉, I번째 싸이클의 부분곱이 +A 또는 +2A일 때 I+1번째 싸이클의 부분곱은 +2A가 될 수 없고, 또한 I번째 싸이클의 부분곱이 -A 또는 -2A일 때 I+1번째 싸이클의 부분곱은 -2A가 될 수 없다. 아런 관계를 이용하면, 인접한 두 싸이클의 SEL_PP[1:0]사이에서 나타나는 평균 해밍 디스턴스가 가장 작은 값이 되도록 SEL_PP[1:0]를 부호화할 수 있다. 따라서, 평균 해밍 디스턴스가 작아진 만큼 동작상의 전력소모를 줄인다. 이를 구현하는 방법은 다음과 같다.

<107> 부분곱의 값 +A를 선택하는 SEL_PP[1:0]의 값과 +2A를 선택하는 SEL_PP[1:0]의 값이 서로 비트반전 관계가 되도록 하고, 부분곱의 값 -A를 선택하는 SEL_PP[1:0]의 값과 -2A를 선택하는 SEL_PP[1:0]의 값이 서로 비트반전 관계가 되도록 한다.

<108> 앞서 보인 표 2를 참조하면, 부분곱의 값 +A를 선택하는 SEL_PP[1:0]의 값은 "00"이고, +2A를 선택하는 SEL_PP[1:0]의 값은 "11"이다. 또한, 부분곱의 값 -A를 선택하는 SEL_PP[1:0]의 값은 "10"이고, -2A를 선택하는 SEL_PP[1:0]의 값은 "01"이다.

<109> 이와같은 SEL_PP[1:0]의 부호화 방식을 사용하면, I번째 싸이클의 부분곱이 +2A이고 I+1번째 싸이클의 부분곱이 +A인 경우와 I번째 싸이클의 부분곱이 -2A이고 I+1번째 싸이클의 부분곱이 -A인 경우에서만 SEL_PP[1:0]의 해밍 디스턴스가 2이고 나머지 경우는 모두 SEL_PP[1:0]의 해밍 디스턴스가 1 또는 0이다.

<110> 상술한 실시예에서 보인 본 발명의 수단 또는 방법에 준하여 본 발명의 기술분야에서 통상의 지식을 가진 자는 본 발명의 범위내에서 본 발명의 변형 및 응용이 가능하다. 예를 들면, 도 2 내지 도 3에 보인 모듈러스 재부호화기 및 부스 재부호화기의 회로들은 관련된 표 1 내지 표 2에 보인 진리표에 맞도록 설계함에 있어서 다양한 논리 조합들이 가능할 것이다.

【발명의 효과】

<111> 전술한 본 발명의 실시예에 의하면, 본 발명은 모듈로곱셈 연산장치에서 전력소모를 줄이는 효과가 있다.

<112> 특히, 모듈러스곱과 부분곱의 인가시점을 동기화시킴으로써, 불필요한 신호천이 또는 글리치를 억제함으로싸 동작상의 전력소모를 줄이는 이점이 있다.

<113> 또한, 본 발명은 2-스테이지의 파이프라인 동작을 실현함으로써 동작주파수를 향상시킬 수 있는 이점이 있다.

【특허청구범위】

【청구항 1】

소정주파수의 클럭에 응답하며 제1피연산자와 제2피연산자 및 제3피연산자를 이용하여 소정의 결과값을 누산기를 통하여 구하는 연산장치에 있어서:

상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하여 상기 누산기로 전송하는 제1경로;

상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하여 상기 누산기로 전송하는 제2경로;

상기 제1경로를 제어하는 제1신호들을 발생하는 제1선택회로;

상기 제2경로를 제어하는 제2신호들을 발생하는 제2선택회로; 그리고

상기 제1신호들이 발생될 때까지 상기 제2신호들을 상기 클럭에 응답하여 소정시간동안 저장하는 수단을 구비함을 특징으로 하는 연산장치.

【청구항 2】

제1항에 있어서,

상기 제1신호들과 상기 제2신호들이 상기 클럭에 응답하여 상기 제1경로와 상기 제2경로에 동시에 각각 인가됨을 특징으로 하는 연산장치.

【청구항 3】

제2항에 있어서,

상기 제1경로가, 상기 제1신호들 중 하나에 응답하여 상기 제1연산값들 중 하나를 선택하는 제1멀티플렉서와, 상기 제1신호들 중 다른 하나에 응답하여 상기 제1멀티플렉서의 출력을 상기 누산기로 전송하는 제1게이트회로를 구비함을 특징으로 하는 연산장치.

【청구항 4】

제2항에 있어서,

상기 제2경로가, 상기 제2신호들 중 하나에 응답하여 상기 제2연산값들 중 하나를 선택하는 제2멀티플렉서와, 상기 제2신호들 중 다른 하나에 응답하여 상기 제2멀티플렉서의 출력을 상기 누산기로 전송하는 제2게이트회로를 구비함을 특징으로 하는 연산장치.

【청구항 5】

제4항에 있어서,

상기 제2신호들 중 상기 하나의 신호가 상기 클럭의 싸이클마다 평균 해밍 디스턴스가 최소인 복수의 비트들로 구성됨을 특징으로 하는 연산장치.

【청구항 6】

제2항에 있어서,

상기 제1신호들 및 제2신호들이 상기 클럭에 응답하여 파이프라인 방식으로 구동됨을 특징으로 하는 연산장치.

【청구항 7】

제6항에 있어서,

상기 제1 및 제2연산값들이 상기 제1 및 제2신호들에 각각 응답하여 파이프라인 방식으로 구동됨을 특징으로 하는 연산장치.

【청구항 8】

소정주파수의 클럭에 응답하며 제1피연산자와 제2피연산자 및 제3피연산자를 이용하여 소정의 결과값을 누산기를 통하여 구하는 연산장치에 있어서:

상기 제1피연산자의 1의 보수값을 포함하는 제1연산값들 중 하나를 선택하는 제1멀티플렉서;

상기 제1멀티플렉서의 출력을 상기 누산기로 전송하는 제1게이트회로;

상기 제2피연산자의 1의 보수값을 포함하는 제2연산값들 중 하나를 선택하는 제2멀티플렉서;

상기 제2멀티플렉서의 출력을 상기 누산기로 전송하는 제2게이트회로;

상기 제1멀티플렉서를 제어하는 제1신호와 상기 제1게이트회로를 제어하는 제2신호를 발생하는 제1선택회로;

상기 제2멀티플렉서를 제어하는 제3신호와 상기 제2게이트회로를 제어하는 제4신호를 발생하는 제2선택회로;

상기 클럭에 응답하여 상기 제1 및 제2신호를 상기 제1멀티플렉서 및 상기 제1게이트회로에 인가하는 제1 및 제2래치들;

상기 클럭에 응답하여 상기 제3 및 제4신호를 저장하는 제1 및 제2플립플럽;

상기 클럭에 응답하여 상기 제1 및 제2플립플럽의 출력들을 상기 제2멀티플렉서 및 상기 제2게이트회로에 인가하는 제3 및 제4래치를 구비하며;

상기 제1 및 제2신호가 발생될 때까지 상기 제3 및 제4신호가 상기 클럭에 응답하여 상기 제1 및 제2플립플럽에 각각 저장됨을 특징으로 하는 연산장치.

【청구항 9】

제8항에 있어서,

상기 제1 및 제3신호가 상기 클럭에 응답하여 상기 제1 및 제2멀티플렉서에 동시에 각각 인가됨을 특징으로 하는 연산장치.

【청구항 10】

제8항에 있어서,

상기 제2 및 제4신호가 상기 클럭에 응답하여 상기 제1 및 제2게이트회로에 동시에 각각 인가됨을 특징으로 하는 연산장치.

【청구항 11】

제8항에 있어서,

상기 제1플립플럽의 출력이 상기 제2선택회로의 입력으로 궤환되어 상기 제3신호의 발생에 반영됨을 특징으로 하는 연산장치.

【청구항 12】

제8항에 있어서,

상기 클럭에 응답하여 상기 제1신호를 저장하는 제3플립플럽을 더 구비함을 특징으로 하는 연산장치.

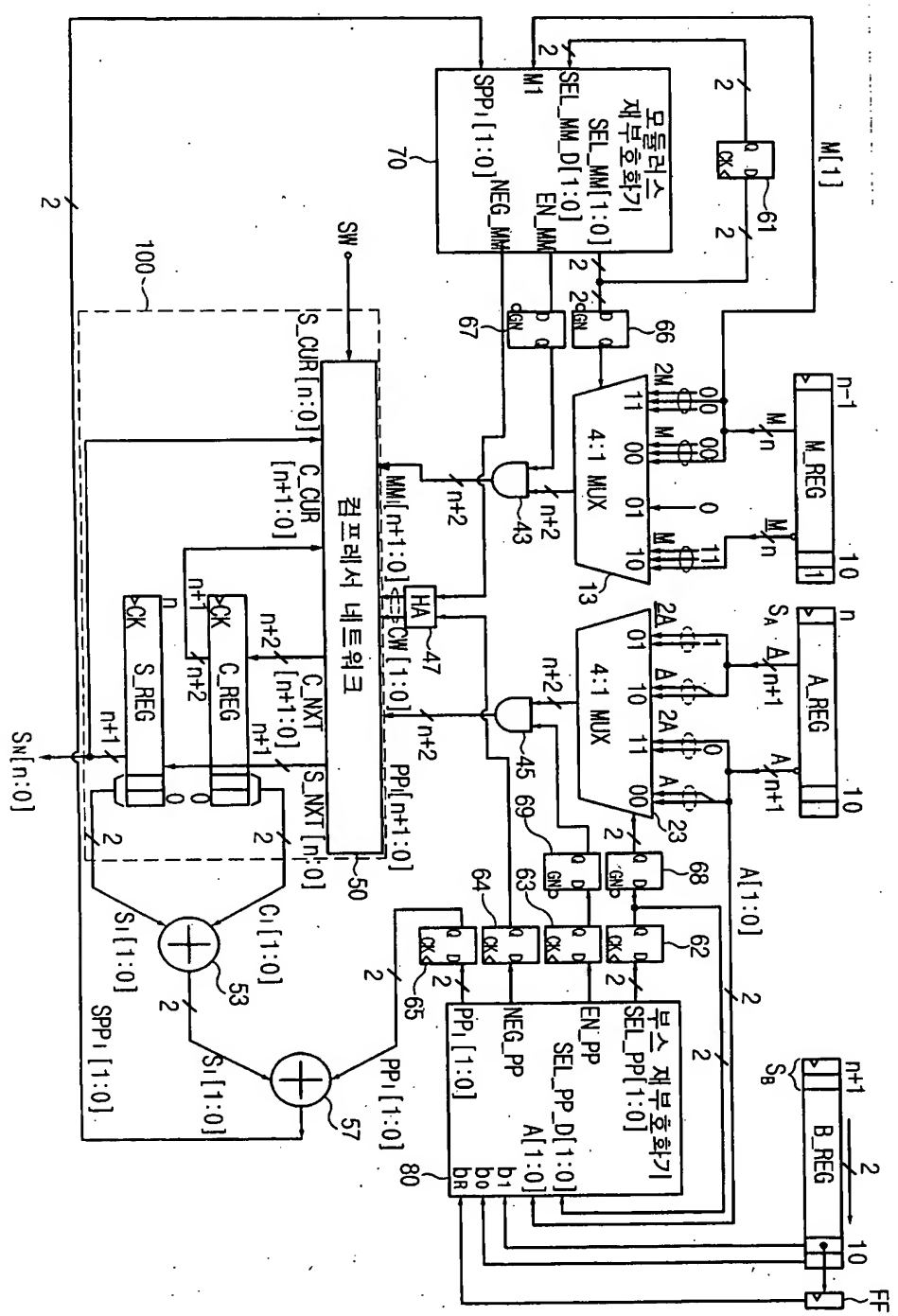
【청구항 13】

제12항에 있어서,

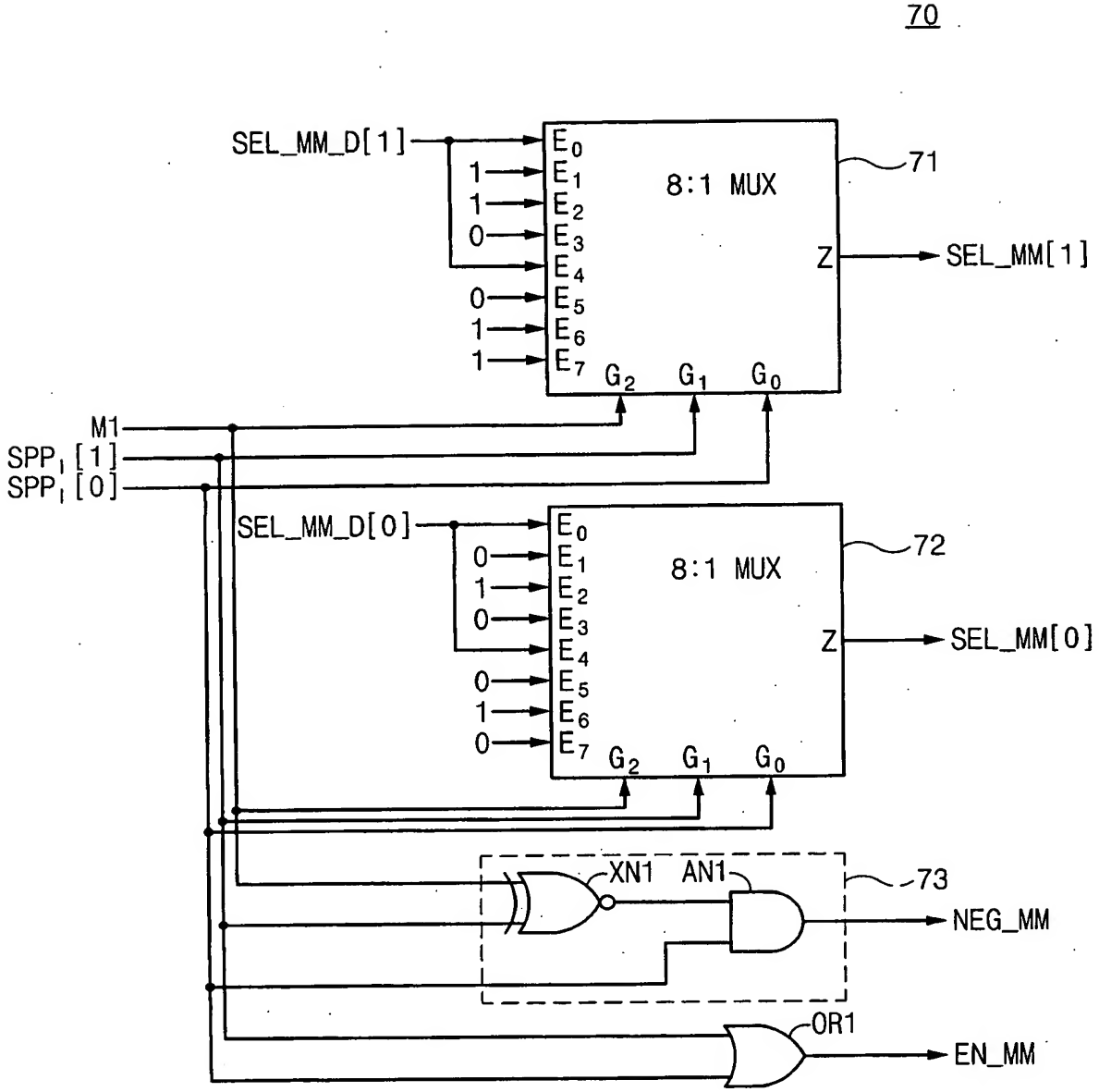
상기 제3플립플럽의 출력이 상기 제1선택회로의 입력으로 궤환되어 상기 제1신호의 발생에 반영됨을 특징으로 하는 연산장치.

【도면】

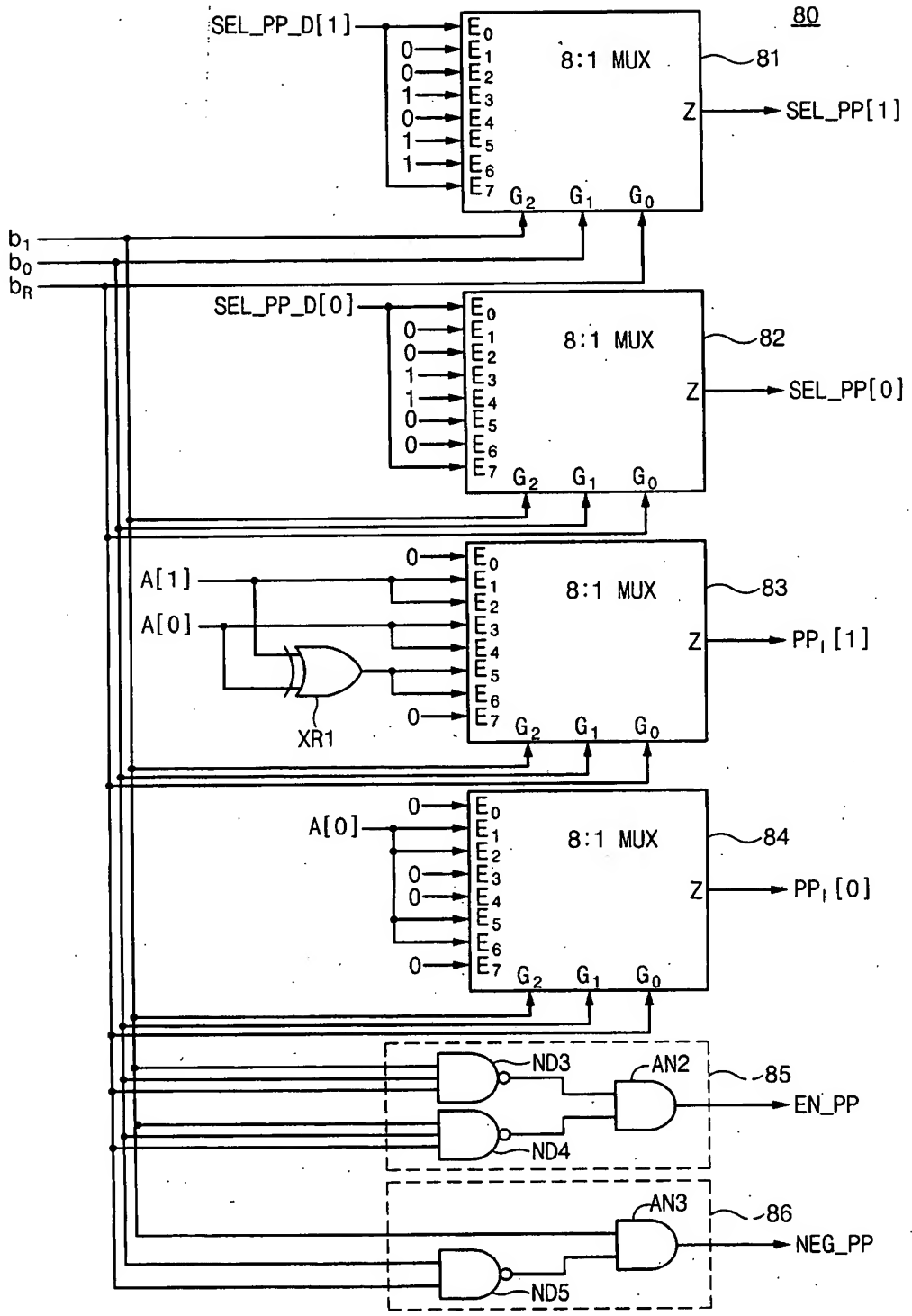
【도 1】


M[1]
M_REG
A_REG
B_REG
FF
모듈러스 재부호화기
SEL_MM[1:0]
SEL_MM_D[1:0]
EN_MM
NEG_MM
SPP1[1:0]
4:1 MUX
컴프레서 네트워크
C_CUR [n:0]
C_NXT [n+1:0]
S_NXT [n:0]
S_REG
C_REG
HA
FECW [1:0]
PP1[n+1:0]
SW
SN[n:0]
부스 재부호화기
SEL_PP[1:0]
SEL_PP_D[1:0]
EN_PP
NEG_PP
PP1[1:0]
A[1:0]
SA
SB
S1[1:0]
C1[1:0]

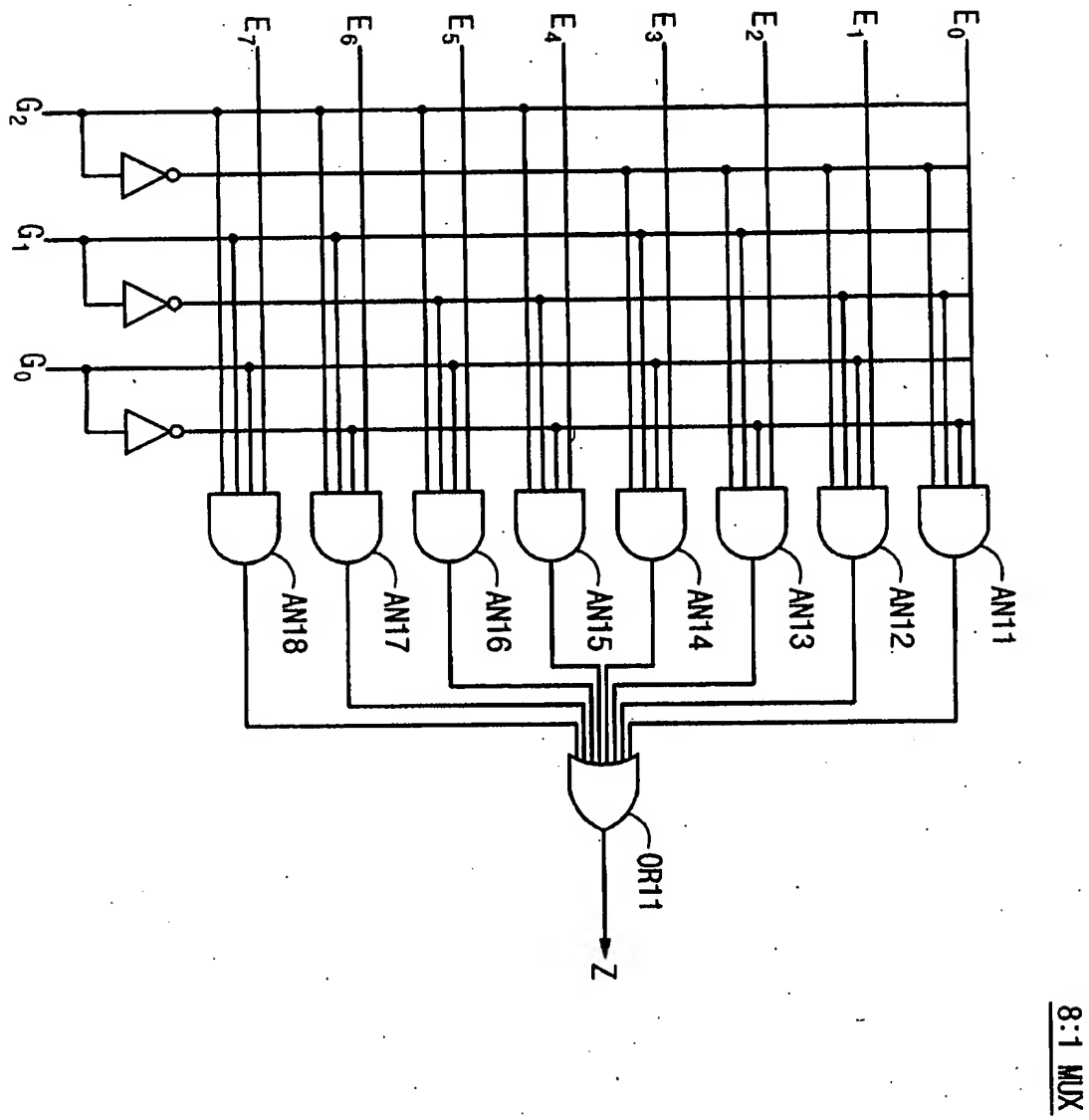
【도 2】


70
SEL_MM_D[1]
1
1
0
0
1
1
E0
E1
E2
E3
E4
E5
E6
E7
8:1 MUX
71
Z
SEL_MM[1]
G2
G1
G0
M1
SPPI[1]
SPPI[0]
SEL_MM_D[0]
0
1
0
0
1
0
8:1 MUX
72
SEL_MM[0]
XN1
AN1
73
NEG_MM
OR1
EN_MM

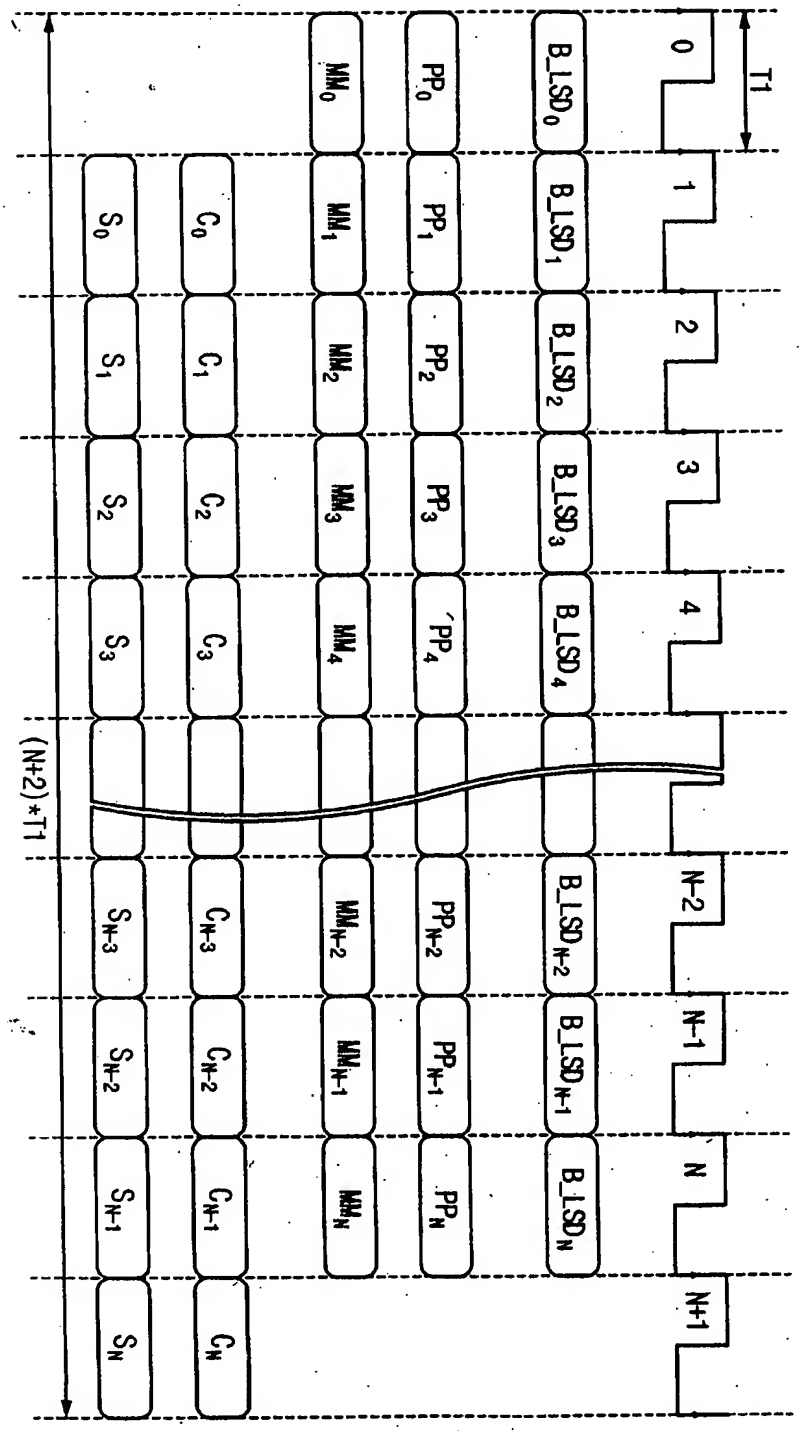
【도 3】


80
SEL_PP_D[1]
8:1 MUX
81
Z
SEL_PP[1]
0
0
1
0
1
1
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7
G_2 G_1 G_0
b_1
b_0
b_R
SEL_PP_D[0]
8:1 MUX
82
Z
SEL_PP[0]
0
0
1
1
0
0
A[1]
A[0]
XR1
8:1 MUX
83
Z
PP_I [1]
A[0]
8:1 MUX
84
Z
PP_I [0]
ND3
AN2
85
EN_PP
ND4
AN3
86
NEG_PP
ND5

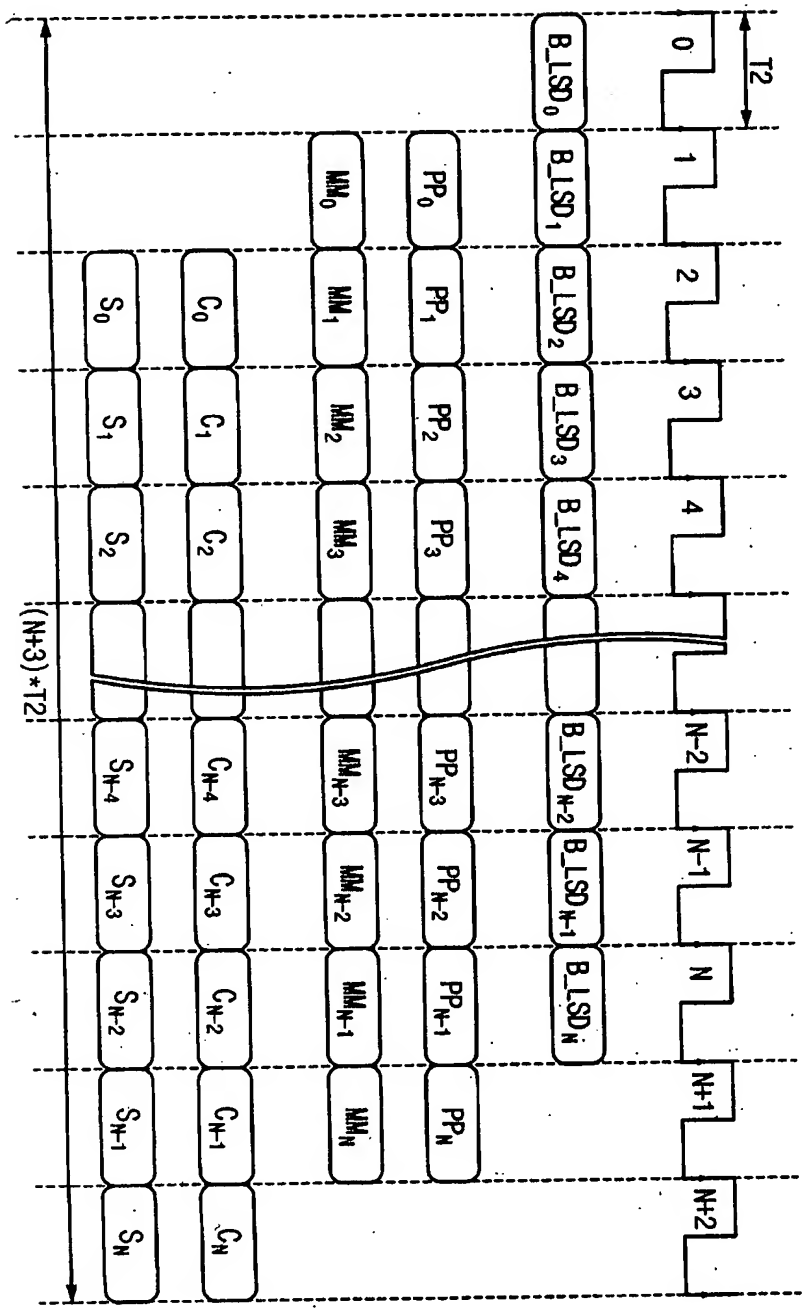
【도 4】


E7
E6
E5
E4
E3
E2
E1
E0
G2
G1
G0
AN18
AN17
AN16
AN15
AN14
AN13
AN12
AN11
OR11
Z
8:1 MUX

【도 5】


0
1
2
3
4
N-2
N-1
N
N+1
T1
(N+2)*T1
B_LSD0
B_LSD1
B_LSD2
B_LSD3
B_LSD4
B_LSDN-2
B_LSDN-1
B_LSDN
PP0
PP1
PP2
PP3
PP4
PPN-2
PPN-1
PPN
MM0
MM1
MM2
MM3
MM4
MMN-2
MMN-1
MMN
S0
S1
S2
S3
SN-3
SN-2
SN-1
SN
C0
C1
C2
C3
CN-3
CN-2
CN-1
CN

【도 6】


T2
0
1
2
3
4
N-2
N-1
N
N+1
N+2
((N+3)*T2)
B_LSD0
B_LSD1
B_LSD2
B_LSD3
B_LSD4
B_LSDN-2
B_LSDN-1
B_LSDN
PP0
PP1
PP2
PP3
PPN-3
PPN-2
PPN-1
PPN
MM0
MM1
MM2
MM3
MMN-3
MMN-2
MMN-1
MMN
S0
S1
S2
SN-4
SN-3
SN-2
SN-1
SN
C0
C1
C2
CN-4
CN-3
CN-2
CN-1
CN